NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVICE PROFILING ANALYSIS IN DEVICE-AWARE NETWORK

by

Shang-Yuan Tsai

December 2004

Thesis Advisor:	Singh Gurminder
Thesis Co-Advisor:	John Gibson

Approved for public release, distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE December 2004	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: Device Profiling Analysis in Device-Aware Network		5. FUNDING NUMBERS	
6. AUTHOR(S) Shang-Yuan Tsai			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (*maximum 200 words*)

As more and more devices with a variety of capabilities are Internet-capable, device independence becomes a big issue when we would like the information that we request to be correctly displayed. This thesis introduces and compares how existing standards create a profile that describes the device capabilities to achieve the goal of device independence.

After acknowledging the importance of device independence, this paper utilizes the idea to introduce a Device-Aware Network (DAN). DAN provides the infrastructure support for device-content compatibility matching for data transmission. We identify the major components of the DAN architecture and issues associated with providing this new network service. A Device-Aware Network will improve the network's efficiency by preventing unusable data from consuming host and network resources. The device profile is the key issue to achieve this goal.

14. SUBJECT TERMS CC/PP, UPnP, SyncML, RDF, DevInf			15. NUMBER OF PAGES 83
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release, distribution is unlimited

DEVICE PROFILING ANALYSIS IN DEVICE-AWARE NETWORK

Shang-Yuan Tsai
Captain, Taiwan Army
B.S., Chung Cheng Institute of Technology, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEM ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2004

Author: Shang-Yuan Tsai

Approved by: Singh Gurminder
               Thesis Advisor

               John Gibson
               Thesis Co-Advisor

               Dan C. Boger
               Chairman, Department of Information Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As more and more devices with a variety of capabilities become Internet-enabled, device independence becomes a big issue when we would like the information that we request to be correctly displayed. This thesis introduces and compares how existing standards create a profile that describes the device capabilities to achieve the goal of device independence.

After acknowledging the importance of device independence, this thesis utilizes the idea to introduce a Device-Aware Network (DAN). DAN provides the infrastructure support for device-content compatibility matching for data transmission. We identify the major components of the DAN architecture and issues associated with providing this new network service. A Device-Aware Network will improve the network efficiency by preventing unusable data from consuming host and network resources. The device profile is the key to achieve this goal.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Singh Gurminder, who gave me a lot of supports not only on my thesis work but also the courses that I took from him. I would also like to thank John Gibson for readily agreeing to be my Co-Advisor. His academic opinions help a lot on how to organize my thesis.

Next, I would like to thank my government authorities for giving me this opportunity to study abroad. Last but not the least, I would like to thank the colleagues from my office who shared my work for two years during my absence.

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

A.   BACKGROUND

A key challenge to communications on the network-centric battlefield is that the end-devices must utilize limited resources to support the mission operations. This requires the devices to conserve resources by avoiding the reception, transmission and processing of unusable information, a capability not currently available. Today's networks are completely unaware of the capability of their end-points. Being dumb pipes, they cannot optimize traffic to match the capabilities and requirements of their end devices. For example, when a very large image is delivered to a small handheld device, a significant amount of network resource is wasted because the handheld device is unable to store or display the image. This situation becomes serious in Web applications. When more and more Internet-capable devices are available, different devices have different display specifications. In order to display the Web content correctly on different displays, the authors have to create the Web pages in different versions in parallel, which is impractical.

With such a problem present, the World Wide Web Consortium (W3C) introduces the concept of device independence [1]. The idea is to let the client send the request with information associated with the end device. The purpose of this information is to provide information that may be needed to allow the final response to be repurposed to the capabilities of the client. The request and the delivery context flow from the client, through any intermediaries to the server. The server can use the appropriate repositories of content in constructing the response to the request.

The goal of a Device-Aware Network (DAN) is to match the information delivered to the capability of the end device, thereby optimizing the network resource usage. On the battlefield, all resources -- including time, network bandwidth and battery capacity -- are very limited. A device-aware network avoids the waste that happens in current device-ignorant networks. By eliminating unusable traffic, a device-aware network reduces the time the end-devices spend receiving extraneous information, and thus saves time and conserves battery life.

To efficiently transmit information on a device-aware network, the capabilities and conditions of an end-device must be defined in advance to adapt the data format. As in the example above, when we want to transfer an image that can be displayed appropriately in a handheld device screen, the resolution of the display has to be pre-defined. Therefore, the concept of a device profile is utilized in device-aware networks. In this protocol, the specification of device hardware and software can be described systematically. However, some features of a device are not always static and need to be updated periodically, such as device position, power status, bandwidth, and temperature etc. If the original device profile is not kept current with these dynamic features, communication errors may happen and the performance of the whole network may be influenced, especially in resource constrained networks. As a result, dynamically delivering device profiles is necessary in a device-aware network.

B.   APPROACH

This thesis will discuss the current standards for device profiling and give a comparison of these standards. It will then identify a suitable standard that can serve as the starting point for creating a device profile request scenario for a device-aware network. Currently the available standards for device profiling are: Composite Capabilities/Preference Profile (CC/PP) developed by the W3C, User Agent Profile (UAProf) developed by the WAP forum, SyncML developed by the mobile technology industry and Universal Plug and Play (UPnP) developed by Microsoft. These standards are developed for dealing with device independence by specifying device capabilities in a device profile. The primary utility of these standards is in content repurposing so different end devices can efficiently utilize the limited bandwidth.

C.   THESIS ORGANIZATION

This thesis is organized into the following chapters:

Chapter I: Introduction. Describe the need for device profiles along with an overview of device-aware network and the available standards.

Chapter II: Device Profiles in a Device-Aware Network. Give an introduction on the architecture of a device-aware network and the basic requirements to achieve device independence.

Chapter III: Device Profile Standards. Provide an overview of currently applied standards with respect to device profiling. Then give a comparison of these standards and find a suitable standard that can be applied to the device-aware network as an initial demonstration.

Chapter IV: Device Profile Creation. Give a detailed description of device profile created from the conclusions of Chapter III.

Chapter V: Conclusion. Give a conclusion based on the thesis work and provide recommendations for future work on device-aware networks.

THIS PAGE INTENTIONALLY LEFT BLANK

II.   DEVICE PROFILE IN DEVICE-AWARE NETWORK

A.   DEVICES

Due to device proliferation, content providers can no longer deliver one version of their content to the user because they need to deliver an appropriate form of content depending on the capabilities of the viewing device. Re-authoring content to support different markup languages or the different capabilities of each device is clearly impractical, while providing content for a single device or browser excludes large numbers of users.

Users want to view Internet content and use web applications on a variety of devices, including PCs, electronic book readers, PDAs, phones, interactive TVs, voice browsers, printers and embedded devices such as cameras. A useful summary of typical variations in device capabilities is shown in Figure 1 [2].
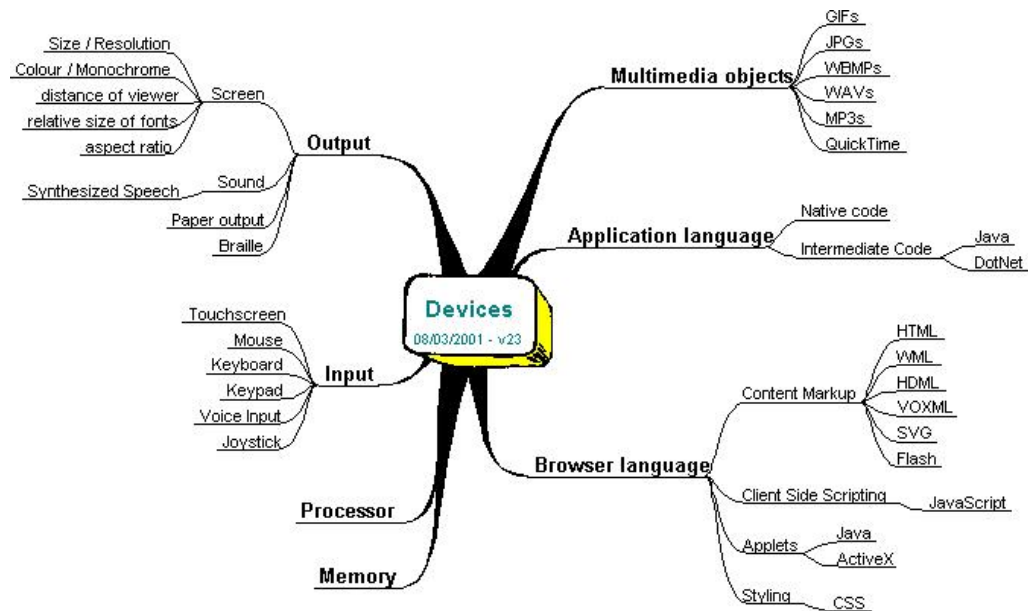
Figure 1.   Variations in Device Capabilities [From: 3]

When the device uses content, it receives it in the form of multimedia objects, application languages or browser languages (shown on the right hand of Figure 1). Current devices support a variety of different content types partly determined by their underlying hardware capabilities (shown on the left hand side of Figure 1). In order to support device independence we must be able to deliver content in a format compatible

with a device. For example, if a handheld device can read GIF images but not JPEG images, it is necessary to convert one format to another. In addition, the content must reflect the underlying hardware capabilities of the device so we may need to do some additional image processing if the target device can only display a 240x340 resolution image properly [2].

B.   DEVICE-AWARE NETWORKS INTRODUCTION

Based on the discussion of previous section, to achieve the goal of device capabilities matching for data transmission and content repurposing, a new high-level network service can be introduced. The design of a Device-Aware Network (DAN) is for such a purpose. A Device-Aware Network improves the network efficiency by preventing unusable data from consuming host and network resources. While DAN is useful in a wired environment, it can be especially beneficial in wireless and mobile environments where network as well as end-host device resources are scarce [3].

C.   ARCHITECTURE OVERVIEW

In a device-aware network, however, the network is required to perform more than the best-effort delivery service; it will need to optimize traffic to match the capabilities and needs of end devices. As a result, our design considers the use of device profiles as an integral part of the network architecture, not just a new addition to the basic protocols as an afterthought. Thus, we do not constrain the DAN design to the Internet architecture; all network components necessary to make the network aware of end device are considered [3].

D.   SERVICES IN A DEVICE-AWARE NETWORK

Device-Aware Networks provide two main services. The first service relates to the sharing of device profile and capability information on the network. Therefore, information exchange and discovery protocols must be developed to support the device-awareness in the network. DAN supports encapsulating device information along with each packet transmitted, allowing any network nodes (e.g.,, routers and end systems) to access the device information in order to perform DAN-related processing. For lack of a

6

better term, we will refer to DAN-related operations as being done at the DAN layer, as if DAN is a separate service layer between the network and transport services. To build device-aware features into the existing IP networks, DAN services may be viewed as a new layer in the network architecture, or an extension to current network layer services. However, our proposed DAN design does not fit the conventional network layering scheme, since its network service is also application dependent. [3]

E. USING DEVICE PROFILE IN DEVICE-AWARE NETWORK

There are three different ways of content repurposing by specifying the device capabilities in a device profile, as shown in Figure 2: the server, the proxy and the client browser. If adaptation occurs at the server or the proxy, these entities will need to know something about the capabilities of the end device. They will either need a unique identifier for the client device so they can retrieve a capability specification from a repository, or they will need the capability specification itself.
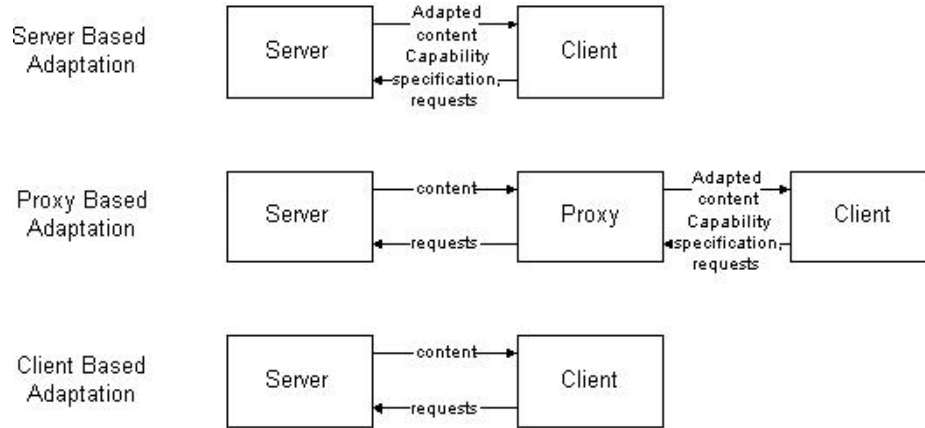
Figure 2.   Content Repurposing Types [From: 2]

Currently, servers and proxies can determine the identity of a particular device using the request header field in the HTTP protocol. In addition there are four alternative proposed capability specification schemes: the W3C composite capability / preferences profile (CC/PP), the Wireless Application Group (WAG) User Agent Profile (UAPROF) standard, the SyncML Device Information standard (DevInf) and the Universal Plug and Play Standard (UPnP). Each of these standards will be discussed in later chapters.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DEVICE PROFILE STANDARDS

A. CC/PP AND UAPROF

1. CC/PP Introduction

A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. As the number and variety of devices connected to the Internet grows, there is a corresponding increase in the need to deliver content that is tailored to the capabilities of different devices. Some limited techniques, such as HTTP "accept" headers and HTML "alt=" attributes, already exist. As part of a framework for content adaptation and contextualization, a general-purpose profile format is required that can describe the capabilities of a user agent and preferences of its user. CC/PP is designed to be such a format.

CC/PP is based on RDF, the Resource Description Framework, which was designed by the W3C as a general-purpose metadata description language. RDF was designed to describe the metadata or machine-understandable properties of the Web. RDF is a natural choice for the CC/PP framework since device profiles are metadata intended primarily for communication between end devices and resource data providers [4].

2. RDF Introduction

RDF is based on the idea of identifying things using Web identifiers (Uniform Resource Identifier - URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. To make this discussion somewhat more concrete as soon as possible, the group of statements "there is someone whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr." could be represented as the RDF graph in Figure 3 [4].

Figure 3.   An RDF Graph Describing Eric Miller [From: 4]

Figure 3 illustrates that RDF uses URIs to identify:

- individuals, e.g., Eric Miller, identified by http://www.w3.org/People/EM/contact#me
- kinds of things, e.g., Person, identified by http://www.w3.org/2000/10/swap/pim/contact#Person
- properties of those things, e.g., mailbox, identified by http://www.w3.org/2000/10/swap/pim/contact#mailbox
- values of those properties, e.g., mailto@w3.org as the value of the mailbox property (RDF also uses character strings such as "Eric Miller" as the values of some properties)

RDF also provides an XML-based syntax (called RDF/XML) for recording and exchanging these graphs. Figure 4 is a small chunk of RDF in RDF/XML corresponding to the graph in Figure 3:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

<contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
</contact:Person>

</rdf:RDF>
```
Figure 4.   RDF/XML Describing Eric Miller [From: 4]

Like HTML, this RDF/XML is machine processable, and, using URIs, can link pieces of information across the Web. However, unlike conventional hypertext, RDF URIs can refer to any identifiable thing, including things that may not be directly retrievable on the Web (such as the person Eric Miller). The result is that in addition to describing such things as Web pages, we can also describe the characteristics or capabilities of an Internet accessable device [4].

3.   CC/PP Architecture

a.   CC/PP Profile Structure
A CC/PP profile is broadly constructed as a 2-level hierarchy:

- a profile having at least one or more components, and

- each component having at least one or more attributes.

The initial branches of the CC/PP profile tree describe major components of the client. Examples of major components are:

- the hardware platform upon which software is executing,

- the software platform upon which all applications are hosted, or

- an individual application, such as a browser [4].

b.   LC/PP Profile Component Attribute
A CC/PP profile describes client capabilities and preferences in terms of a number of "CC/PP attributes" for each component.

The description of each component is a sub-tree whose branches are the capabilities or preferences associated with that component. Though RDF makes modeling a wide range of data structures possible, including arbitrary graphs, complex data models are usually best avoided for profile attribute values. A capability can often be described using a small number of CC/PP attributes, each having a simple, atomic value. Where more complex values are needed, these can be constructed as RDF subgraphs. One useful case for complex attribute values is to represent alternative values; e.g., a browser may support multiple versions of HTML. A hypothetical profile might look like Figure 5 [4]:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
        xmlns:ex="http://www.example.com/schema#">

  <rdf:Description
      rdf:about="http://www.example.com/profile#MyProfile">

    <ccpp:component>
      <rdf:Description
       rdf:about="http://www.example.com/profile#TerminalHardware">
        <rdf:type
          rdf:resource="http://www.example.com/schema#HardwarePlatf
          orm" />
        <ex:displayWidth>320</ex:displayWidth>
        <ex:displayHeight>200</ex:displayHeight>
        </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
       rdf:about="http://www.example.com/profile#TerminalSoftware">
        <rdf:type
          rdf:resource="http://www.example.com/schema#SoftwarePlatf
          orm" />
        <ex:name>EPOC</ex:name>
        <ex:version>2.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
        </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
       rdf:about="http://www.example.com/profile#TerminalBrowser">
        <rdf:type
          rdf:resource="http://www.example.com/schema#BrowserUA" />
        <ex:name>Mozilla</ex:name>
        <ex:version>5.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
        <ex:htmlVersionsSupported>
```

```
            <rdf:Bag>
              <rdf:li>3.2</rdf:li>
              <rdf:li>4.0</rdf:li>
            </rdf:Bag>
          </ex:htmlVersionsSupported>
        </rdf:Description>
      </ccpp:component>

    </rdf:Description>
  </rdf:RDF>
```

Figure 5.   Complete CC/PP profile example in XML [From: 4]

c.   CC/PP Profiles Defaults

Each component of a device profile may indicate a single separate resource that in turn indicates a subordinate collection of default attribute values. This collection of default values can be a separate RDF document that is named via a URI, or it can appear in the same document as the client profile (though, in practice, there is probably little value in defaults in the same document). If an attribute in the collection of defaults is also present in the main part of the client profile, the non-default value takes precedence. The intent is that a hardware vendor or system supplier may provide default values that are common to a number of systems in a place easily accessible to an origin server, and then use the device profile to specify variations from the common profile. The owner of the device or system operator may be able to add or change options, such as additional memory, that add new capabilities or change the values of some original capabilities [4].

d.   CC/PP Exchange Protocol

The major disadvantage of this format is that it is verbose. Some networks, such as mobile phone networks, are very slow, and this would add an overhead to handle the device profiles. There are several optimizations possible to help deal with network performance issues. One strategy is to use references (URIs). Instead of enumerating each set of attributes, a reference can be used to name a collection of attributes, such as the hardware platform defaults. This has the advantage of enabling the separate fetching and caching of functional subsets. Another problem is to propagate changes in the current CC/PP descriptions to an origin server or a proxy. One solution is to transmit the entire

CC/PP descriptions with each change. This is not ideal for slow networks. An alternative is to send only the changes. The "CC/PP exchange protocol" is proposed to deal with this situation. The CC/PP exchange protocol does not depend on the profile format which it conveys. Therefore, another profile format besides the CC/PP description format could be applied to the CC/PP exchange protocol.

The strategy of the CC/PP exchange protocol is to send a request with profile information using as few references as possible. For example, a user agent issues a request with URIs which address the profile information, and if the user agent changes the value of an attribute, such as turning sound off, only that change is sent together with the URIs. When an origin server receives the request, the origin server inquires CC/PP repositories for the CC/PP descriptions, using the list of URIs. Then the origin server creates a tailored content using the fully enumerated CC/PP descriptions.

The origin server might not obtain the fully enumerated CC/PP descriptions when any one of the CC/PP repositories is not available. In this case, it depends on the implementation whether the origin server should respond to the request with a tailored content, a non-tailored content or an error. In any case, the origin server should inform the user agent of the fact. A warning mechanism has been introduced for this purpose [5].

4.   UAPROF
a.   *UAProf Introduction*

UAProf is a Wireless Application Protocol (WAP) Forum specification that is designed to allow wireless mobile devices to declare their capabilities to data servers and other network components. The design of UAProf is already based on RDF. As such, its vocabulary elements use the same basic format that is used for CC/PP.

In this specification, UAProf considers five different categories of device capability, as shown in Figure 4: software, hardware, browser, network and WAP. This means the server can adapt to the capabilities of the network as well as the capabilities of the device [2].

14
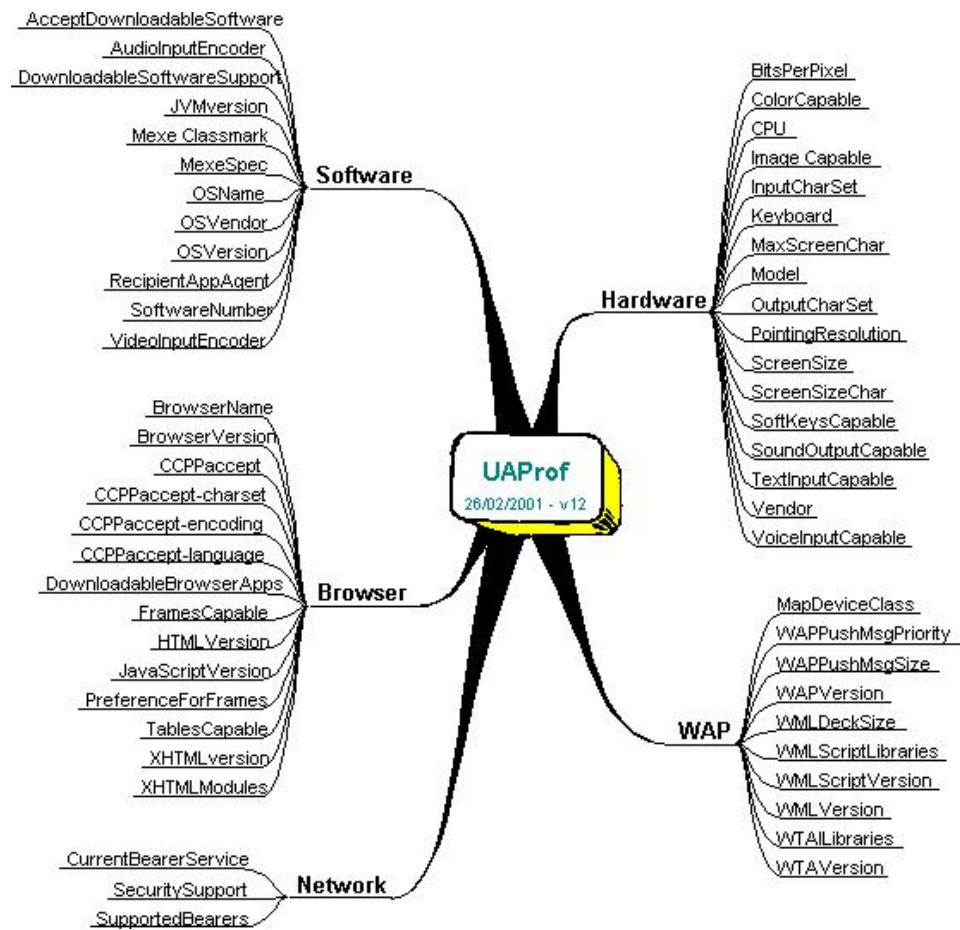
Figure 6.   The UAProf specification [From: 2]

b.   UAProf Architecture

The UAProf is an End-to-End framework, as shown in Figure 7. The information of the end device is collected on the client device, encoded into an efficient binary form, transmitted and cached within a Wireless Session Protocol (WSP) session, optionally enhanced with information provided with a particular request, optionally combined with other information available over the network, and made available to the origin server. Over the Internet, this specification assumes the use of the CC/PP and the CC/PP Exchange Protocol over HTTP.

15
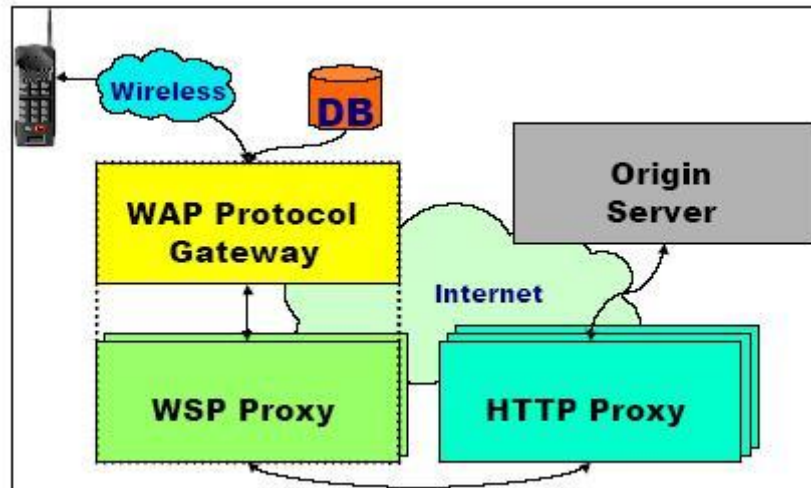
Figure 7.   UAPROF End-to-End framework [From: 6]

The End-to-End framework consists of five components:

- A client device capable of requesting and rendering WAP content.

- A wireless network employing WAP 1.1 or later protocols.

- A WAP-capable gateway capable of translating WAP protocol requests into corresponding requests over the Internet and translating responses from the Internet into corresponding responses over the WAP protocols.

- The Internet or an intranet using TCP/IP-based protocols and possibly having one or more protocol gateways and Web/HTTP proxies.

- An origin (Web) server that can generate requested content.

Though this specification refers to five end-to-end system components, actual configurations may physically deploy those components in many forms. For example, the latter three components (WAP gateway, Internet/intranet, and origin server) might easily be merged into a single server-side system connected to the WAP network. Moreover, the WAP gateway may itself be distributed, with different hosts serving as endpoints for different layers of the WAP protocol stack [6].

c.   *Client Device*

The device profile consists of information gathered from the device hardware, active user agent software, and user preferences. In many cases, much of this information must be pre-installed directly on the device, possibly in the firmware. For instance, the device may publish a single URI that points to default device capability information made available by the device manufacturer. Similarly, the user agent may publish a single URI that points to default software information made available by the software developer.

The client device is assumed to employ the WAP communications protocols, particularly WSP, to request content from an origin server. The device profile is transmitted and maintained using designated WSP headers in accordance with that specification . This information is initially conveyed when a WSP session is established with a compliant WAP protocol gateway. The client thereafter assumes that the WAP gateway caches the device profile and will apply it on all requests initiated during the lifetime of the WSP session [6].

d.   *Wireless Network and WAP Gateway*

WSP sessions are carried over wireless networks that are capable of implementing the WAP protocols. The WAP gateway represents the server-side endpoint for the client's WSP session. To support these sessions, the gateway must suport the Wireless Datagram Protocol (WDP) and Wireless Transaction Protocol (WTP) layers . As part of its WSP session implementation, the WAP gateway must implement WSP header caching, thereby allowing it to hold the device profile conveyed by the client device during session establishment. The WAP gateway is responsible for translating WSP requests into appropriate HTTP requests for delivery over an intranet or the Internet to the designated origin server. In forwarding these requests, the gateway must also forward the current device profile associated with the session and/or request. This specification requires that the gateway use the HTTP Extension Framework to convey the device profile within HTTP headers, as discussed above regarding the CC/PP protocol. When generating the HTTP request, the gateway may optionally augment the received

device profile with additional data obtained from local databases, such as a network Home Location Register (HLR).

The WAP gateway is also responsible for translating HTTP responses into appropriate WSP responses for delivery over the wireless network to the requesting end device. In forwarding these responses, the gateway must also forward any device profile usage headers provided by the origin server and/or any intermediate HTTP proxies [6].

e. Internet or Intranet

The HTTP requests generated by the WAP gateway are conveyed over an intranet or the Internet, capable of carrying TCP/IP-based requests and responses. In passing through these networks, the request may pass through one or more proxies, each responsible for forwarding the request toward the particular origin server designated in the request. These proxies may conform to either the HTTP 1.0 or HTTP 1.1 protocol standards. It is important to note that the HTTP 1.0 proxies will discard all device profiles contained in the HTTP request. The HTTP 1.1 proxies may or may not forward the device profile intact, depending on whether the information is conveyed in mandatory or optional headers. For HTTP 1.1 proxies that are aware of the HTTP Extension Framework and CC/PP Exchange Protocol over HTTP optionally may add information to the device profile conveyed in the outbound HTTP request.

Internet network elements, both proxies and origin servers, may provide content caching capabilities. Caching is complicated by the presence of device profile because the content associated with a particular URI may differ according to the device presented to the origin server. As a rule, therefore, an HTTP proxy or origin server will only deliver content from its cache if both of the following conditions hold:

- The content has not expired from the cache, in accordance with standard HTTP caching semantics.
- The device profile associated with the cached request exactly matches the device profile associated with the new request.

To minimize the possibility that an intermediate proxy that is unaware of CC/PP accidentally sources content from its cache without first checking for a matching

18

CC/PP profile, an origin server may set the Cache-Control headers in the HTTP response to prevent the proxy from doing any caching [6].

f.   *Origin Server*

The origin server is the ultimate recipient of the request initiated by the end device (and forwarded as an HTTP request from the WAP gateway). The origin server is responsible for receiving the request and generating appropriate content that is subsequently transported as an HTTP response to the WAP gateway. In generating this response, the origin server extracts the device profile conveyed with the HTTP request, resolves all indirect references to information stored at other repositories in the network, if necessary, and uses that information to select or otherwise customize the content being delivered to the client. In generating the HTTP response using the CC/PP Exchange Protocol over HTTP, a server must indicate the extent to which the device profile was honored in producing the content contained within the HTTP response [6].

B.   UPNP

1.   UPnP Introduction

With the addition of Device Plug and Play (PnP) capabilities to the operating system it became a great deal easier to setup, configure, and add peripherals to a PC. Universal Plug and Play (UPnP), which was developed by Microsoft, extends this simplicity to include the entire network, enabling discovery and control of devices, including networked devices and services, such as network-attached printers, Internet gateways, and consumer electronics equipment.

With UPnP, a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices—all automatically; truly enabling zero configuration networks. Devices can subsequently communicate with each other directly; thereby further enabling peer-to-peer networking.

UPnP uses standard TCP/IP and Internet protocols, enabling it to seamlessly fit into existing networks. Using these standardized protocols allows UPnP to benefit from a wealth of experience and knowledge, and makes interoperability an inherent feature. Because UPnP is a distributed, open network architecture, defined by the protocols used,

it is independent of any particular operating system, programming language, or physical medium (just like the Internet). UPnP does not specify the APIs applications will use, allowing operating system vendors to create the APIs that will meet their customers' needs [7].

2. UPnP Architecture

There are three basic components of a UPnP network: devices, services and control points. Figure 6 is the block diagram of the three components.
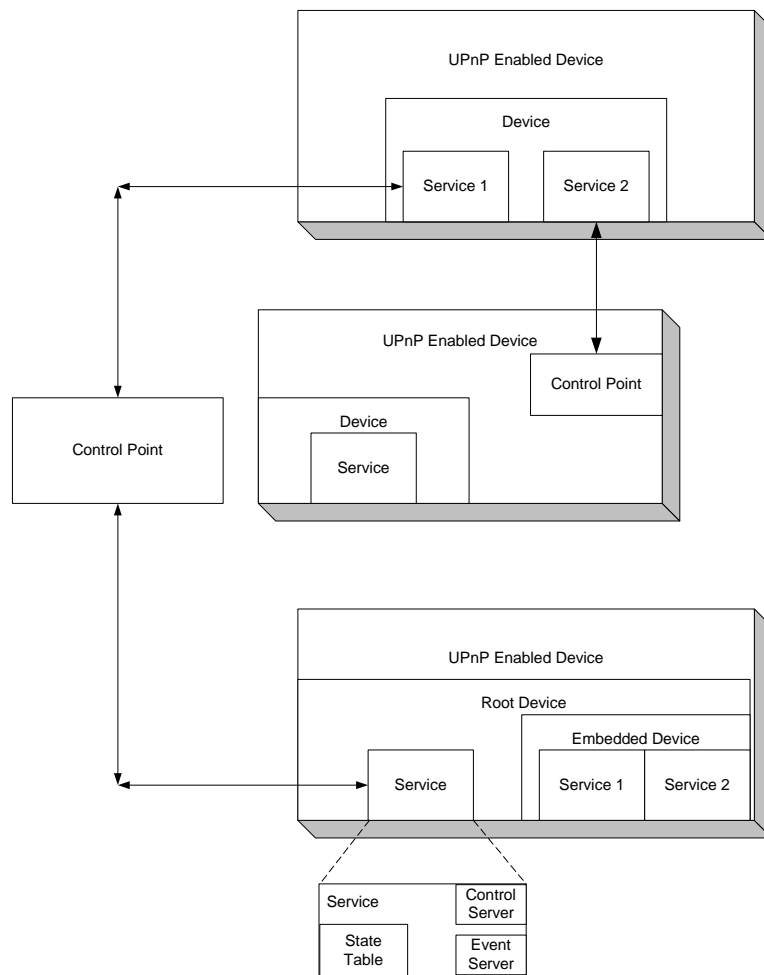
Figure 8.   UPnP Control Points, Devices, and Services [From: 7]

a. UPnP Devices

A UPnP device is a container of services and nested devices. For instance, a VCR device may consist of a tape transport service, a tuner service, and a clock service. A TV/VCR combo device would consist not just of services, but a nested device as well.

Different categories of UPnP devices will be associated with different sets of services and embedded devices. For instance, services within a VCR will be different than those within a printer. Consequently, different working groups will standardize on the set of services that a particular device type will provide. All of this information is captured in an XML device description document that the device must host. In addition to the set of services, the device description also lists the properties (such as device name and icons) associated with the device [7].

b. UPnP Services

The smallest unit of control in a UPnP network is a service. A service exposes actions and models its state with state variables. For instance, a clock service could be modeled as having a state variable, current_time, which defines the state of the clock, and two actions, set_time and get_time, which allow designers to control the service. Similar to the device description, this information is part of an XML service description standardized by the UPnP forum. A pointer (URL) to these service descriptions is contained within the device description document. Devices may contain multiple services.

A service in a UPnP device consists of a state table, a control server and an event server. The state table models the state of the service through state variables and updates them when the state changes. The control server receives action requests (such as set_time), executes them, updates the state table and returns responses. The event server publishes events to interested subscribers anytime the state of the service changes. For instance, the fire alarm service would send an event to interested subscribers when its state changes to "ringing" [7].

c. *UPnP Control Points*

A control point in a UPnP network is a controller capable of discovering and controlling other devices. After discovery, a control point could:

- Retrieve the device description and get a list of associated services.

- Retrieve service descriptions for interesting services.

- Invoke actions to control the service.

- Subscribe to the service's event source. Anytime the state of the service changes, the event server will send an event to the control point.

It is expected that devices will incorporate control point functionality (and vice-versa) to enable true peer-to-peer networking [7].

d. *Protocols Used by UPnP*

UPnP leverages many existing, standard protocols. Using these standardized protocols aids in ensuring interoperability between vendor implementations. Figure 7 is the protocol stack of UPnP.

| UPnP Vendor Defined |
| UPnP Forum Working Committee Defined |
| UPnP Device Architecture Defined |

| SSDP | HTTPMU (Discovery) | GENA | SSDP | HTTPU (Discovery) | SOAP (Control) | HTTP | | | | HTTP (Description) | GENA (Events) |

| UDP | TCP |

| IP |
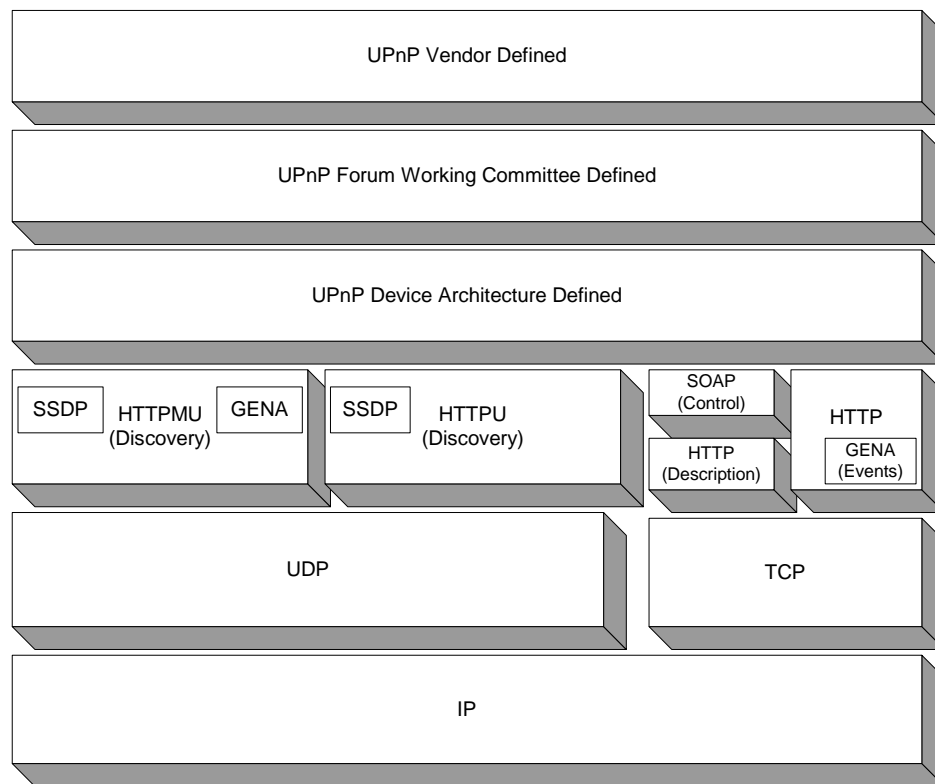
Figure 9.   The UPnP protocol stack [From: 7]

3.   Activities Involved in UPnP Network

a.   *Addressing*

The foundation for UPnP networking is the TCP/IP protocol suite, and the key to this suite is addressing. Each device must have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device is first connected to the network. If a DHCP server is available, the device must use the IP address assigned to it. If no DHCP server is available, the device must use Auto IP to get an address. In brief, Auto IP defines how a device intelligently chooses an IP address from a set of reserved private addresses, and is able to move easily between managed and unmanaged networks.

A device may implement higher layer protocols outside of UPnP that use friendly names for devices. In these cases, it becomes necessary to resolve friendly host (device) names to IP addresses. Domain Name Services (DNS) are usually used for this.

A device that requires or uses this functionality may include a DNS client and may support dynamic DNS registration for its own name to address mapping [7].

b. Discovery

Once devices are attached to the network and addressed appropriately, discovery can take place. Discovery is handled by the Simple Service Discovery Protocol (SSDP). When a device is added to the network, SSDP allows that device to advertise its services to control points on the network. When a control point is added to the network, SSDP allows that control point to search for devices of interest on the network.

The fundamental exchange in both cases is a discovery message containing a few essential specifics about the device or one of its services, for example, its type, identifier, and a pointer to its XML device description document [7].

c. Description

The next step in UPnP networking is description. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message.

Devices may contain other, logical devices and services. The UPnP description for a device is expressed in XML and includes vendor-specific manufacturer information including the model name and number, serial number, manufacturer name, URLs to vendor-specific Web sites, and so forth. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation [7].

d. Control

After a control point has retrieved a description of the device, the control point has the essentials for device control. To learn more about the service, a control point must retrieve a detailed UPnP description for each service. The description for a service is also expressed in XML and includes a list of the commands, or actions, the

service responds to, and parameters or arguments for each action. The description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics.

To control a device, a control point sends an action request to a device's service. To do this, a control point sends a suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using Simple Object Access Protocol (SOAP). In response to the control message, the service returns action specific values or fault codes [7].

e. Eventing

A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information.

The service publishes updates by sending event messages. Event messages contain the names of one of more state variables and the current value of those variables. These messages are also expressed in XML and formatted according to the Generic Event Notification Architecture (GENA). A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service. To support multiple control points, all subscribers are sent all event messages, subscribers receive event messages for all evented variables, and event messages are sent no matter why the state variable changed (in response to an action request or due to a state change) [7].

f. Presentation

If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a browser, and depending on the capabilities of the page, allow a user to control the device and/or view device status. The degree to which each of these can be accomplished depends on the specific capabilities of the presentation page and device [7].

C.   SYNCML
1.   SyncML Introduction

SyncML was developed by the mobile technology industry. It is a specification for a common data synchronization framework and XML-based format, or representation protocol, for synchronizing data on networked devices. SyncML can also be used for peer-to-peer data synchronization. SyncML is specifically designed to handle the case where the network services and the device store the data they are synchronizing in different formats or use different software systems.

The framework is depicted in Figure 10. In the figure, the scope of the SyncML framework is shown by the dotted-line box. The Framework consists of SyncML representation protocol, as well as a conceptual SyncML Adapter and the SyncML Interface [8].
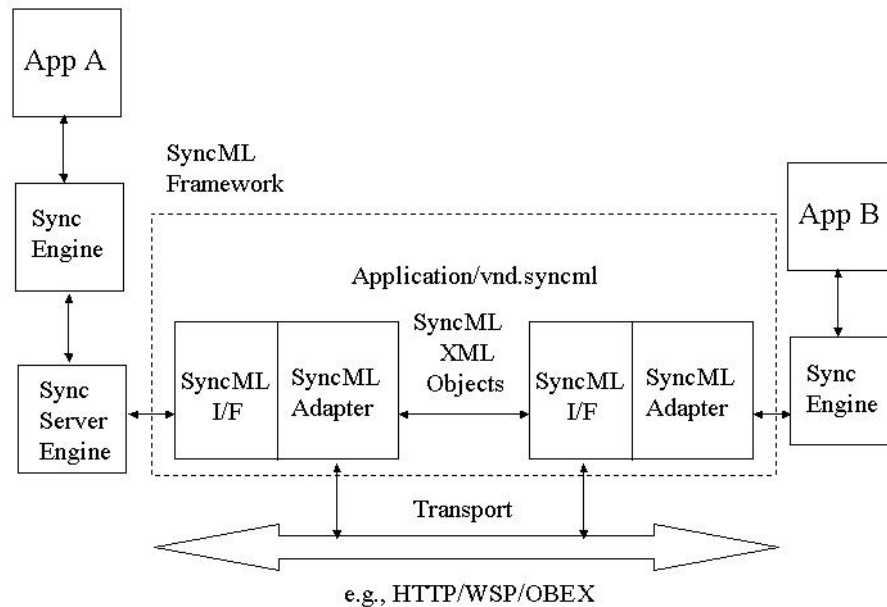
Figure 10.   SyncML framework [From: 8]

The application "A" depicts a networked service that provides data synchronization with other applications, in this case application "B," on some networked device. The service and device are connected over some common application layer protocols, such as HTTP and WSP. Application "A" utilizes a data synchronization

26

protocol, implemented as the "Sync Engine" process. The data synchronization protocol is manifested on the network by client applications accessing the "Sync Server" network resource. The "Sync Server Agent" manages the "Sync Engine" access to the network and communicates the data synchronization operations to/from the client application. The "Sync Server Agent" performs these capabilities through invocations to functions in the "SyncML I/F" or interface. The "SyncML I/F" is the application programming interface to the "SyncML Adapter". The "SyncML Adapter" is the conceptual process that the originator and recipient of SyncML formatted objects utilize to communicate with each other. The "SyncML Adapter" is also the framework entity that interfaces with the network transport, which is responsible for creating and maintaining a network connection between Application "A" and Application "B." Application "B" utilizes a "Sync Client Agent" to access the network and it's "SyncML Adapter," through invocations of functions in the "SyncML I/F" [8].

2.   SyncML Packages and Messages

In SyncML, the data synchronization operations are conceptually bound into a SyncML Package . The SyncML Package is just a conceptual frame for one or more SyncML Messages that are required to convey a set of data synchronization semantics. A SyncML Message is a well-formed, but not necessarily valid, XML document. The document is identified by the SyncML root or document element type. This element type acts as a parent container (i.e., root element type) for the SyncML Message. The SyncML Message, as specified before, is an individual XML document. The document consists of a header, specified by the "SynHdr" element type, and a body, specified by the "SyncBody" element type. The SyncML header specifies routing and versioning information about the SyncML Message. The SyncML body is a container for one or more SyncML commands. The SyncML Commands are specified by individual element types. The SyncML Commands act as containers for other element types that describe the specifics of the SyncML command, including any synchronization data or meta-information [8].

### 3.	SyncML Capabilities Exchange

SyncML supports capabilities exchange. Capabilities exchange is the ability of a SyncML Client and Server to determine what device, user and application features each supports. The capabilities exchange, from the SyncML Server perspective, is achieved by using the "Get" command to retrieve the device information, user information and application information documents from the SyncML Client. The capabilities exchange, from the SyncML Client perspective, is achieved by using the "Get" command to retrieve the analogous documents from the SyncML Server. These documents contain profile information about support for well-defined features. In addition, the "Put" command can be used by the SyncML Client to push capabilities exchange information to the SyncML Server. The capabilities exchange can also be used to establish or administer SyncML data synchronization services between a SyncML Client and Server [8].

### 4.	Data Identifier Mapping

SyncML does not require that two data stores being synchronized be of the same schema (i.e., aren't homogeneous). Specifically, SyncML allows for both the data identifiers and the data formats to be different in the two data collections. However, in such cases in order to use SyncML, the synchronizing applications would need to provide a mapping between data identifiers in one data store and those in another. For example, a document on the data synchronization server could be identified with a 16-byte, globally unique identifier (GUID). The corresponding version of this document on a mobile device could be identified by a small, two-byte local unique identifier (LUID). Hence, to synchronize the data on the mobile device with the data on the data synchronization server, the synchronizing application would have to map the smaller identifiers of the mobile device to the larger identifiers used by data synchronization server; and vice versa. SyncML includes the necessary mechanism to specify such an identifier mapping [8].

### 5.	Refreshing Data

In addition to synchronization, SyncML includes commands that are not normally thought of as synchronization operations, but are still required in a practical data

synchronization protocol. For example, SyncML provides the capability for refreshing the entire data on the SyncML client with the equivalent synchronization data on the SyncML server. This may be necessary if the SyncML client and the SyncML server versions are no longer "in sync" with each other due to a hardware or power failure in the mobile device, or if the version on the SyncML client has become corrupted or erased from memory. This capability is provided by the SyncML client issuing a "refresh" Alert command to the SyncML server [8].

6.   DevInf Introduction

By the SyncML standard, when two devices are in the process of synchronization, they have to exchange their device profiles in advance. The device profile is exchanged by using the SyncML Device Information (DevInf) standard. The DevInf is represented in a markup language defined by WAP Binary XML (WBXML). In WBXML, the binary format is designed to allow for compact transmission with no loss of functionality or semantic information. It is also designed to preserve the element structure of XML, allowing a browser to skip unknown elements or attributes. The binary format encodes the parsed physical form of an XML document, i.e., the structure and content of the document entities. Meta-information, including the document type definition and conditional sections, is removed when the document is converted to the binary format [9]. Using DevInf, the device profile comes in four parts, as shown in Figure 11.
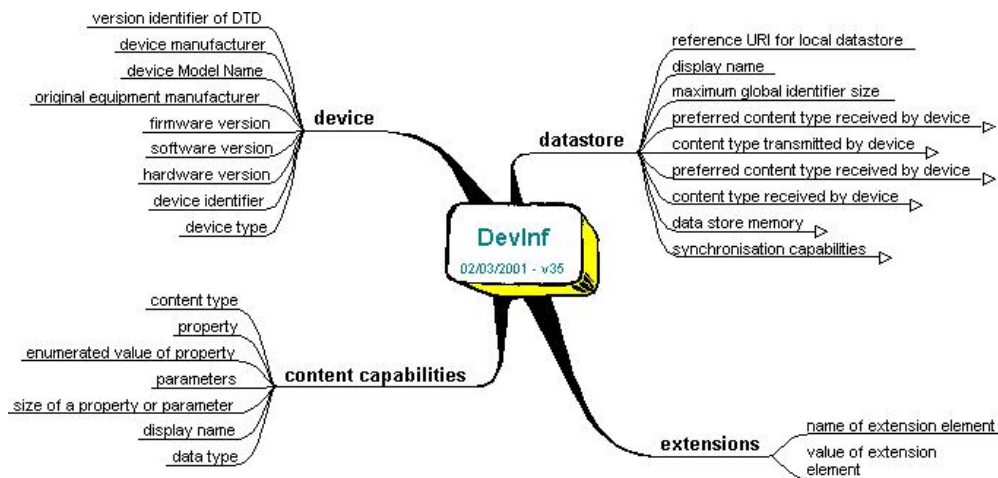
Figure 11.   SyncML DevInf Specification [From: 2]

29

D.   COMPARISON OF THE STANDARDS

From the previous descriptions of the device profile standards, we give a comparison in Table 1.

The design of CC/PP is backwards compatible with UAProf. The goal is that valid UAProf profiles are also valid CC/PP profiles; however, not all CC/PP profiles are necessarily valid UAProf profiles.

	CC/PP	UAProf	UPnP	SyncML
Proposer	W3C	WAP Forum	Microsoft	Communication Industries
Standard used for device profile creation	RDF	RDF	XML	DevInf
Device profile format	XML	XML	XML	XML
Vocabulary in device profile	User-defined based on application	Designed and developed by WAP forum specifically for wireless application	Provided by vendors	Provided by the proposers of the standard
Protocol used for device profile transmission	HTTP Extension Framework (The CC/PP specification does not impose constraints on transmission protocol)	WSP	HTTP	HTTP/WSP/OBEX
Main application	Content repurposing between end devices and servers	Content repurposing between end devices (mainly wireless devices) and servers	Multimedia devices control (e.g., DVD player, VCR) and information appliance (IA) control	Synchronizati-on between devices (e.g., PDA and PC, mobile phone and PC)
Flexibility for application design	High, developers can create their own device profile vocabularies	Low, the UAProf can be viewed as an application of CC/PP	Low, the device profile description has to be provided by vendor	Low, the standard is focused on mobile communication

Table 1. Comparison of the standards

THIS PAGE INTENTIONALLY LEFT BLANK

IV.   DEVICE PROFILE CREATION (USING CC/PP)

A.   DESIGN OVERVIEW

From the previous discussion, we can realize that to achieve the flexibility of DAN application design, the CC/PP standard is the optimal solution among the available standards. Furthermore, to reduce the load of a server, we may use the architecture of user agent proxy, as shown in Figure 12, to process device profiles from different clients.
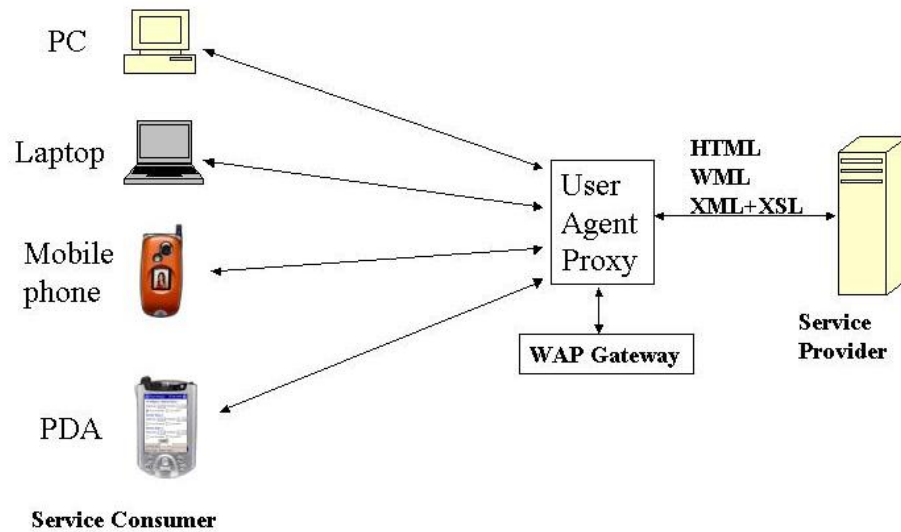
Figure 12.   User Agent Proxy Architecture

For a DAN client, it is necessary to provide the username and password before entering the system. Figure 13 shows the register mechanism for an end-device to login to the DAN system.

Figure 13.   DAN User Registration Mechanism

B.   CC/PP AND UAPROF VOCABULARY

A CC/PP or UAProf vocabulary defines the recognized components, their attributes, and type information. In the CC/PP standard only a few core vocabularies were defined. As for UAProf, there are two versions of vocabulary in the specification [6,10]. When CC/PP was created, it was expected that the creation of multiple vocabularies for device profiles was unavoidable. Therefore, RDF has been designed to cope with data from different sources using different vocabularies [11]. For example, the Intel PCA Developer Network [12] device profile assigns two schemas by using two URIs in its RDF file:

xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330#"
xmlns:pca="http://developer.intel.com/pca/developernetwork/devsupport/pca_schema/2002_01#"

1.   Vocabulary Serialization

A profile can use attributes from multiple vocabularies and schemas. This is called vocabulary serialization. Different vocabularies can be used in a profile using XML namespaces, as shown in a previous example. Within a profile, each vocabulary that is used is associated with an XML namespace which uniquely identifies the attributes in the profile. Because any application or operational environment that uses CC/PP may define its own vocabulary, the vocabularies have to be defined more generally if wider interoperability is taken into consideration [13].

34

2.   Characteristic of Attributes

The attribute definition includes identifying the semantic description, attribute type, and sample values. For a device profile, it is obvious that some attribute values may keep changing, e.g., power status, device temperature, CPU frequency, connection speed. It is necessary to classfy attributes into two catalogs (static and dynamic) if we would like to cooperate with the end devices efficiently. In application design, the resolution rule of a device profile should be assumed to be the default rule for static values. For dynamic attributes, the values shown in a profile are assumed to be initial values only. Subsequently, the values can be updated at a fixed rate based on the actual value at the time the value is sampled. A monitor mechanism on the client side can be created to perform the function of reading the dynamic values and updating the values in the profile periodically [13]. In APPENDIX A, the tables give a detailed description and data types of the device profile vocabularies developed by the Intel PCA Network. APPENDIX B is the RDF format of a device profile derived from APPENDIX A.

3.   Profile Resolution

UAProf and CC/PP standards define a data format and a protocol to be used for the exchange and resolution of device capability information. They do not specify how to collect this information or how to customize content based on the profile information. Profile information must be collected on client devices and resolved on a server [13]. Both CC/PP and UAProf clients may split up profile information in order to send it to the server in an efficient way. They do this by using a standard profile, known as a reference profile, and a list of overrides specific to the requesting device, known as a profile-diff. Other devices in the communication path, such as proxies, may also add profile-diffs. The process of reassembling the final profile from the reference profile(s) and profile-diff(s) is known as profile resolution. CC/PP does not specify the exact mechanism for profile resolution, apart from requiring that default attribute values are always overridden by non-default attribute values. UAProf, in contrast, specifies a set of resolution rules that apply to non-default values. Each attribute in a vocabulary is associated with a specific resolution rule that is applied when multiple attribute values are encountered. In UAProf these resolution rules are order dependent; for example, "locked" means take the first

value encountered whereas "override" means take the last value encountered. Unfortunately, these rules are difficult to implement in RDF, as RDF models do not have any implicit concept of ordering statements. Ordering must be done explicitly, e.g., using an RDF Sequence (rdf:Seq [15]). Unlike RDF, XML does implicitly order elements in documents. When statements in an RDF model are unordered it is impossible to apply the UAProf resolution rules to a single RDF model. Possible solutions include representing each profile or profile-diff as a separate model and keeping track of the order of these models. This allows resolution to be performed between models. An alternative approach is to convert profiles to an intermediate data structure that stores attribute order before performing profile resolution [11].

4.   Validating CC/PP and UAProf Profiles

In order to validate CC/PP and UAProf profiles, there must be a set of rules that determine what constitutes a valid profile. According to the CC/PP Structure and Vocabularies Working Draft [4], a CC/PP profile must meet the following constraint: a profile must be valid XML and a valid XML serialization of RDF. Based on the description, there are two possible solutions for application developers to validate CC/PP and UAProf profiles: XML schema parser and RDF schema parser [16].

a.   Validation Using XML Schema Parser

It is important to note that although RDF schema and XML schema are both schema languages, they perform slightly different roles: RDF schema's primary aim is to provide a machine-readable description of a particular vocabulary rather than provide mechanisms for validating data. XML schema, on the other hand, can be used to validate XML documents and enforce strict structural and datatype constraints. Therefore, one solution to the validation problem in CC/PP would be to use XML schema parser to validate profiles. In order to use XML schema in this way, it is necessary to solve another related problem: in the XML serialization of RDF it is possible to serialize a single RDF graph in several different ways, making the required XML schema complex and unwieldy. The solution proposed here is to use XSLT (XSL Transformation) [19] to convert a profile to a constrained form of RDF that maintains all the information from the

original serialization. After this the profile can be validated using XML schema, to ensure that it is both syntactically correct and that it uses all referenced vocabularies correctly. This process is shown diagrammatically in Figure 14 [16].
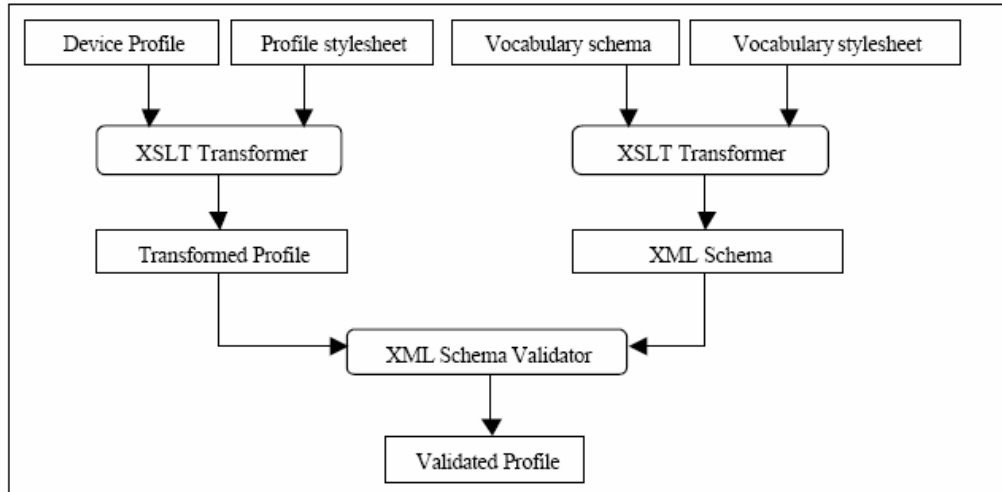
Figure 14.   Validating Device Profiles Using XSLT and XML Schema [From: 16]

Using the stylesheet approach to validate device profiles has a number of advantages: First, it provides a simple mechanism for validation that makes use of existing tools, e.g., XSLT and XML schema. Furthermore, using this functionality in a program is simple, since there are several open source XML schema parsers and XSLT transformers available, such as Apache Xerces and Apache Xalan [18]. It also makes use of existing information, e.g., the RDF schemas for UAProf. The downside of performing validation in this way is that both profiles and vocabularies must be transformed before they can be validated. Ideally, it should be possible to validate profiles without any changes, as validating transformed profiles can lead to error messages that are difficult to interpret, as they refer to a different profile than the one presented by the user.

Secondly, because there are various versions of the UAProf vocabulary, each using a different namespace URI, it is necessary to have separate stylesheets to convert profiles and schema belonging to the different versions. This is due to a restriction in XSLT that prevents stylesheets from inserting namespace declaration attributes into a document [16].

37

b.   ***Validation Using RDF Schema Parser***

Performing validation of RDF documents using a RDF schema parser is more complex than validating XML documents, because there are no standardized tools available to accomplish this task. This approach has the advantage of not requiring any transformations of profiles or schema.

To determine the structure to which profiles must adhere, the validator exploits the two-level structure of UAProf profiles (profiles contain components, which contain properties). The UAProf vocabulary gives regular expressions for the datatypes it defines, and these can be used in the validator. It became apparent, however, that many profiles do not adhere to these specified expressions. For example, the literal datatype has the following regular expression in the schema:

[A-Za-z0-9/.\-_]+

There are many literals in profiles that contain spaces, asterisks, semicolons and various other characters forbidden by this expression. Although this problem is easily solved by extending the expression to allow a wider variety of strings, ideally these regular expressions should be machine readable, rather than written as XML comments, to make it easier for RDF parsers to extract them and use them in profile validation [16].

5.   Device Profiles Serialization in XSLT

For content authors, it is a good solution to simplify the transformation process of information in device profiles by using XML and XSLT. One problem with manipulating CC/PP or UAProf profiles in XSLT is that these device profiles are represented using RDF. Although RDF models can be represented in an XML serialization, it is difficult to manipulate this serialization in XSLT, as it can represent the same model in many different ways [11]. To make the device profiles easier to manipulate, we can create a profile that only consists of profile attributes with all RDF format removed in XSLT, e.g.,:

```
<browser>
        <ScreenSize>90x120</ScreenSize>
        <IsColorCapable>Yes</IsColorCapable>
        <CcppAccept>
           <li>text/html</li>
           <li>text/plain</li>
           <li>image/jpeg</li>
         </CcppAccept>
</browser>
```

6.   Device Profiles Matching Rules

Different Internet-capable devices have different input, output, hardware, software, network and browser capabilities [18]. In order to provide optimized content to different clients, the server must process device profiles correctly. The following stylesheet demonstrates how we can use XPath [19] conditional statements to query profiles within XSLT:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version
  ="1.0">
  <xsl:param name="device-capabilities"/>
  <xsl:template match="/">
    <xsl:if test="contains($device-capabilities/browser/CcppAccept,'wml') and
        contains($deli-capabilites/browser/ScreenSize,'90x120') and
        contains($deli-capabilities/browser/IsColorCapable,'Yes')">
        <wml>
          <card id="init" newcontext="true">
          <p>Color device with 90x120 screen</p>
          </card>
        </wml>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

In this example, the stylesheet only generates a WML page if the device is WML capable, color capable and has a screen size 90x120 pixels. In addition to the contains() function, we can also use the >, >=, <, <=, =, and != expressions in conditional statements. However, the UAProf standard uses various data types that are difficult to process using these conditional statements. First, UAProf has a data type called dimension that consists

of two numbers separated by an "x," e.g., 90x120. It is not possible to apply numerical expressions to this data type, so only the contains() function may be used. Second, numbers in UAProf are integers, so instead of representing version numbers as numbers they are represented as string literals.

In [18] the HP Lab proposed a method called "capability class" to overcome such problems. Capability class works as follows: a number of capability classes are defined where each class is associated with a set of constraints. When a server receives a profile, it evaluates each set of constraints to determine if the target device belongs to one or more of the capability classes. Once it has determined which capability classes are supported by the device, this information is passed to the stylesheet to guide transformation. For example, consider the file shown below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<classes>
 <class name="smallScreen">
  <or>
   <lessthan value="160x160">ScreenSize</lessthan>
   <lessthan value="20x20">ScreenSizeChar</lessthan>
  </or>
 </class>
 <class name="largeScreen">
  <or>
   <greaterthan value="320x240">ScreenSize</greaterthan>
   <greaterthan value="80x40">ScreenSizeChar</greaterthan>
  </or>
 </class>
 <class name="jpegcapable">
  <contains value="image/jpeg">CcppAccept</contains>
 </class>
 <class name="color">
  <true>ColorCapable</true>
 </class>
 <class name="blackandwhite">
  <not>
   <true>ColorCapable</true>
  </not>
 </class>
 <class name="colorphone">
  <and>
   <lessthan value="90x120">ScreenSize</lessthan>
   <contains value="wml">CcppAccept</contains>
```

```
       <true>IsColorCapable</true>
      </and>
    </class>
  </classes>
```

This file defines four capability classes: *smallScreen*, *largeScreen*, *jpegcapable* and *color*. In the case of *smallScreen*, the constraints are that the device has a screen smaller than 160 wide and 160 pixels high or if it has a screen that is smaller than 20 characters wide and smaller than 20 characters high. Alternatively a device meets the *jpegcapable* capability class criteria if it can display the MIME type image/jpeg.

Capability class files can contain three Boolean expressions for aggregating constraints: *and*, *or* and *not*. It provides a number of conditionals: *lessthan*, *lessthanequals*, *greaterthan*, *greaterthanequals*, *equals*, *contains* and *true*. Each conditional is only applicable to specific attribute types, as shown in Table 2. For dimensions, the conditionals mean the result is true if both numbers are met; otherwise it returns false.

Conditional	Compatible UAProf data types
lessthan	number, dimension
lessthanequals	number, dimension
greaterthan	number, dimension
greaterthanequals	number, dimension
equals	number, dimension, single literal
contains	set of literals, sequence of literals
true	boolean

Table 2.   Conditionals of Capability Class

B.   AVAILABLE APPLICATION FOR CC/PP AND UAPROF PROFILING

In [20] there is a list of software available for handling CC/PP and UAProf device profiles. This thesis provides a testing report on one of these software and demonstrates a platform that can handle XML/XSLT architecture.

1.   DELI Introduction

HP Labs' DElivery context LIbrary (DELI) is a toolkit that allows Java servlets to resolve HTTP requests containing delivery context information from CC/PP or UAProf

capable devices and query the resolved profile. It also provides support for legacy devices so that the proprietary delivery context descriptions currently used by applications can be replaced by standardized CC/PP descriptions [22].

2.   Testing Device Profiles in DELI

In order to install DELI and run test servlets, an installation of the Java Runtime Environment with a Java Servlet engine, such as Tomcat are necessary.

a.   *Browser Profiles Testing*

By typing the following address in the Internet Explorer browser, the browser should display the profile properties of Internet Explorer as shown in Figure 15, because the default value is set to reference msie.rdf in the DELI profile directory.

http://localhost:8080/ccpp/html/
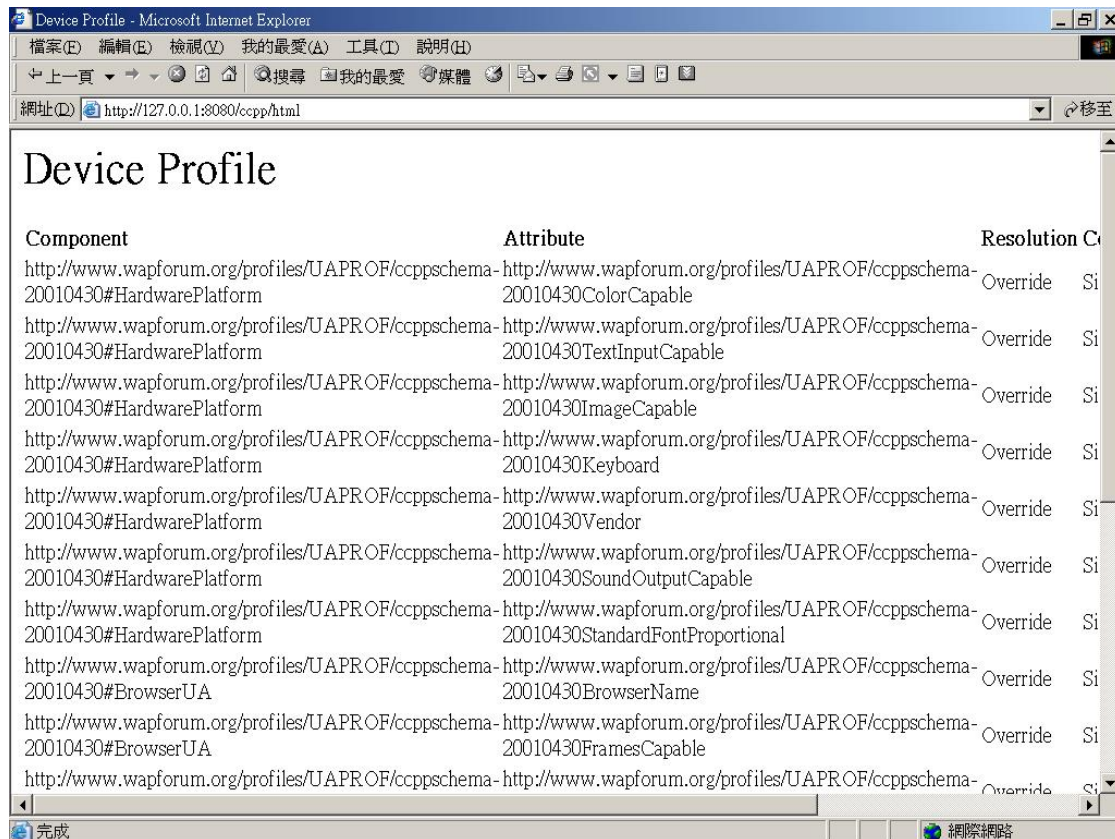
Figure 15.   The Profile of Internet Browser

b. WML Profile Testing

With WML profile testing, we use two simulators to do the simulation: Microsoft Pocket PC 2003 Emulator and Nokia Wap Gateway Simulator. In Microsoft Pocket PC 2003 Emulator, we type the following address and get the device profile result shown in Figure 16.

http://:8080/ccpp/wml/

Figure 16.   WML Profile Testing in Pocket PC Simulator

In Nokia Wap Gateway Simulator, we can assign the device profile for reference. For example, if we use Nokia 9210i as our mobile phone interface, the deivce profile can be assigned by the following two parameters in device settings: x-wap-profile and x-wap-profile-diff. These are the headers of Wireless Profiled HTTP (W-HTTP) [23], a protocol proposed by the WAP Forum to transport the device profile. An example W-HTTP request is shown below :

GET /ccpp/html/ HTTP/1.1
Host: localhost
x-wap-profile:"http://localhost:8080/ccpp/profiles/ Nokia_9210i_WML.rdf,"
"1-Rb0sq/nuUFQU75vAjKyiHw=="
x-wap-profile-diff:1;<?xml version="1.0"?>
 <rdf:RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-
                20010430#">
 <rdf:Description rdf:ID="MyDeviceProfile">
   <prf:component>
    <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccpp
                    schema- 20010426#HardwarePlatform"/>
       <prf:BitsPerPixel>16</prf:BitsPerPixel>
      </rdf:Description>
   </prf:component>
  </rdf:Description>
  </rdf:RDF>

In this request, the profile is referenced via the x-wap-profile line and has the URI:

http://localhost:8080/ccpp/profiles/Nokia_9210i_WML.rdf

After the profile reference, there is a value *1-Rb0sq/nuUFQU75vAjKyiHw==* known as a profile-diff digest. The first part of the profile-diff-digest, 1-, is the profile-diff sequence number. This is used to indicate the order of the profile-diffs and to indicate which profile-diff the profile-diff digest refers to [22]. Therefore, we have to add the parameter values of Nokia 9210i HTTP request headers as shown in Figure 17.

44
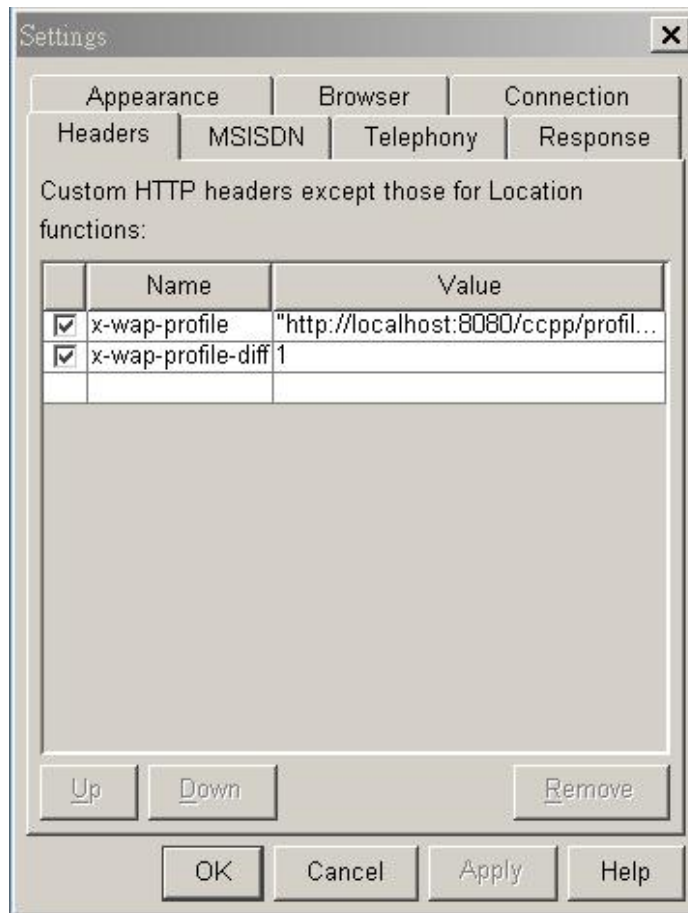
Figure 17. W-HTTP Header Settings

In the Nokia Mobile Browser, if we type *http://localhost:8080/ccpp/wml/* for profile request, then the browser should display the contents of the Nokia 9210i profile, as shown in Figure 18.
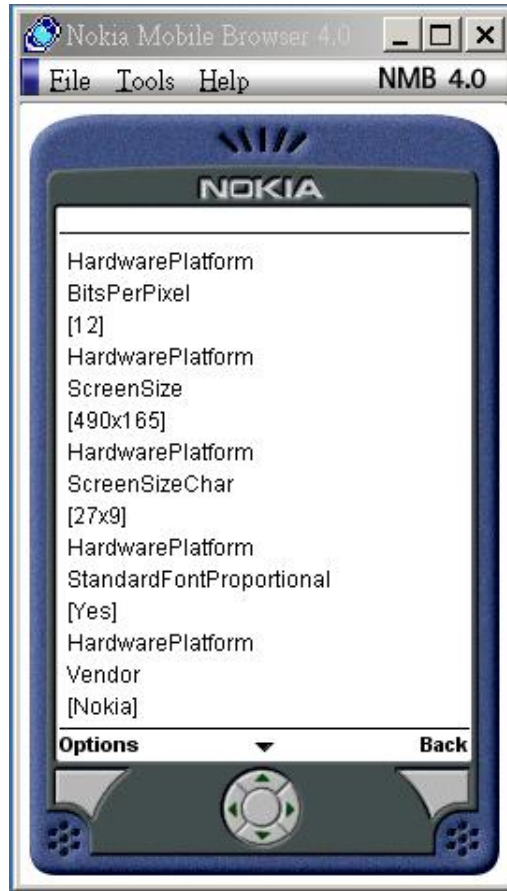
Figure 18.   WML Profile Testing in Mobile Phone Simulator

c.   *Customized Mobile Device Profile*

For most mobile devices, the dynamic parameters of device profiles are not always static, e.g., power status, bandwidth, temperature etc. Therefore, to get the correct information of a device, a detailed dynamic device vocabulary is necessary. In DELI, the new device profile can be created by adding a new device profile schema. To customize a new device profile with dynamic parameters, we use the device profile created by Intel PCA Network and add a new vocabulary (DeviceTemperature) in this schema. For some new mobile devices, the hardware temperature can be monitored by the operating system, which can then turn off the device if the temperature level is higher than the value for device operation. To view the content of the device profile, we use the Java program provided in DELI. By typing the command as shown in Figure 19, the RDF format profile can be resolved and displayed in HTML, formatted as shown in Figure 20.

46

命令提示字元

C:\Tomcat\webapps\ccpp>java com.hp.hpl.deliTest.TestCCPPClient normal http://localhost:8080/ccpp/profiles/pocketpc.rdf output.html

Figure 19.　Resolve Device Profile Command

Device Profile - Microsoft Internet Explorer

檔案(F)　編輯(E)　檢視(V)　我的最愛(A)　工具(T)　說明(H)

上一頁　　　　　　搜尋　我的最愛　媒體

網址(D)　C:\Tomcat\webapps\ccpp\output.html　移至

HTTP/1.1 200 OK Content-Type: text/html Date: Tue, 23 Nov 2004 22:32:00 GMT Server: Apache-Coyote/1.1 Connection: close

Device Profile

Component	Attribute
http://www.wapforum.org/UAPROF/ccppschema-20010330Unknown	http://www.wapforum.org/UAPROF/ccppschema-20010330component
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330ScreenSize
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330Model
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330Vendor
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330CPU
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330InputCharSet
http://www.wapforum.org/UAPROF/ccppschema-20010330#HardwarePlatform	http://www.wapforum.org/UAPROF/ccppschema-20010330ScreenSizeChar
http://www.wapforum.org/UAPROF/ccppschema-	http://www.wapforum.org/UAPROF/ccppschema-20010330BitsPerPixel

完成　　　我的電腦

Figure 20.　HTML Format of Mobile Device Profile

3.　Apache Cocoon Introduction

Apache Cocoon is a web development framework built around the concept of separation of concerns and component-based web development [24]. Cocoon was developed by Apache for publishing XML to multiple target devices. It provides caching to speed up document delivery. It uses XSP, the EXtensible Server Pages Language, an XML compliant version of Java Server Pages to generate XML on the fly.

47

Due to the features of Apache Cocoon, HP Labs now is integrating DELI with Cocoon. By default, the profile resolution function is switched off on the Cocoon website. To turn on the function of profile resolution, we have to add *<map:parameter name="use-deli" value="true"/>* to the pattern match that specifies the stylesheet in sitemap.xmap after the installation of Cocoon. Here is the match used for the deli test stylesheet:

```
<map:match pattern="deli.wml">
    <map:generate src="docs/samples/hello-page.xml"/>
    <map:transform src="stylesheets/deli_test.xsl" type="xslt">
       <map:parameter name="use-deli" value="true"/>
    </map:transform>
    <map:serialize type="wml"/>
 </map:match>
```

Then we can test the profile resolution function by typing the following address to resolve the web browser. The result is shown in Figure 21.

http://localhost:8080/cocoon/deli.html
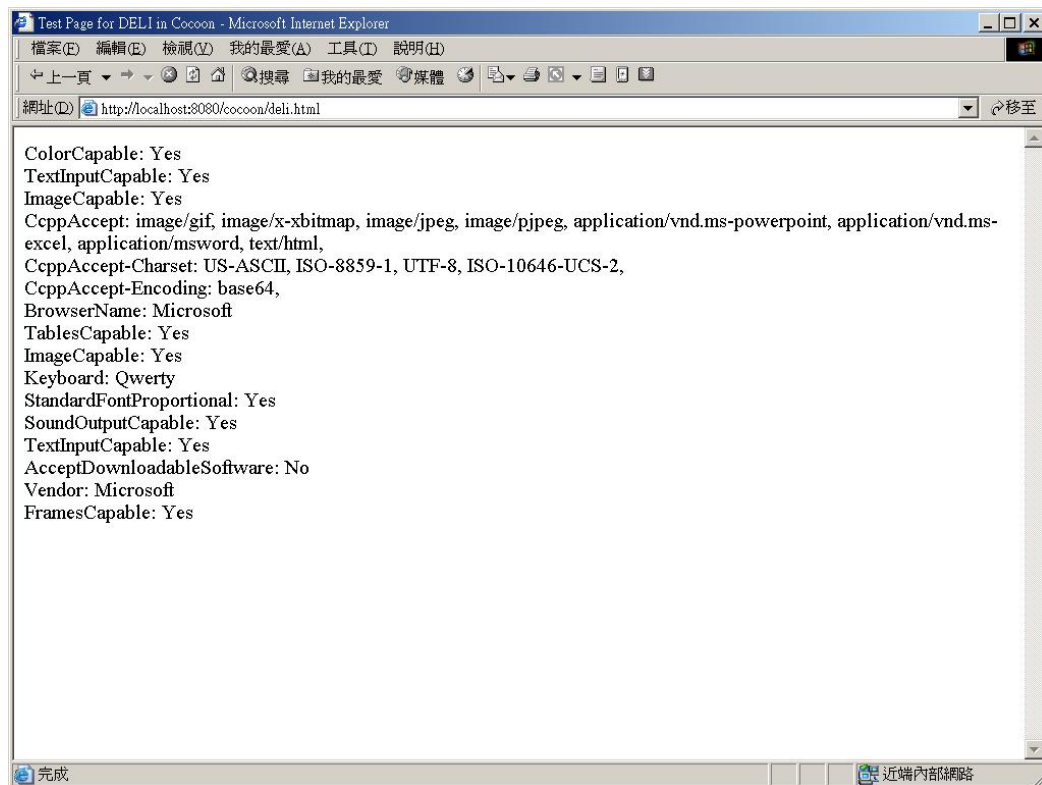

Figure 21.   Browser Profile Resolution in HTML

48

The test result of resolve profile via WML, by typing the following address, is shown in Figure 22.

http://localhost:8080/cocoon/deli.wml

Figure 22.   Browser Profile Resolution in WML

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY

The two main functions of a DAN network are capability discovery and content repurposing. For capability discovery, an alternative design is to exchange device capabilities by creating a device profile for each device. Once the device profiles are known by two end devices, the subsequent control and data exchange can be made. For example, a network printer in DAN can provide the print service by sending a device profile without the need of configuration. Therefore, creating a device profile framework is critical in DAN. This thesis began by providing an overview of the available device profile standards. Then by the comparison of the standards, we concluded that the CC/PP standard is a better choice in implementing DAN network. We demonstrated testing scenarios of device profiles by using DELI and showed the result via HTTP and WML. Even though the CC/PP and UAProf provide a good mechanism for device profiling, there are still some problems that need to be solved, especially for CC/PP. In [4], the W3C lists the issues needed to be considered when developing applications base on the CC/PP standard:

- Device capability exchange protocol

- Trust model between end devices

- Device profile vocabulary

- Security mechanisms

- Constraints on allowable attribute value types

- Attribute value processing and/or matching rules

- Proxy vocabulary and processing

- Rules for request profile identification

- Additional information to be included with any transmitted resource data

- URI forms allowed for identifying referenced profile documents (e.g., defaults)

- Mechanisms for locating and retrieving referenced profile documents

- Interactions with any existing negotiation mechanisms in the host protocol

Upon surveying the current practice in device capability coordination, we concluded that the CC/PP approach provides the most promise for adapting to the DAN architecture. With the specification of UAProf that extended from CC/PP, the vendors can create the device profiles base on the standard before release the reproducts on the market, which solve the problem of interoperability. Furthermore, with the support for delivering device profiles in wireless environment, the developers can easily desgin an application that fits all differetn Internet environment without the problem of compatibility. However, further development needs to be done with respect to the mechanism for dynamic property status reporting and management. A test bed should be developed in order to test the performance characteristics of dynamic profile management over the CC/PP architecture.

B.   FUTURE WORK

As the Device Aware Networking concept is still in its infancy, there are many areas that bear further study. Following are several areas for futher investigation or development.

1.   Content Repurposing

Content repurposing is another function in DANs. After we fetch the hardware and software properties from an end device, the next step is making use of these properties and content adaption. Even though most of the content information can be displayed in HTML format by using a browser as a user interface, there are still other end devices that are not equipped with such software. For example, if we implement DAN on the battlefield, most of the weapon systems console displays do not have the capability to display the web pages. Therefore, displaying the non-HTML content format is another issue needed to be discussed during the process of developing a device-aware network.

2. Creating Legacy Devices Repository

Instead of sending an entire profile with every request, a client can only send a reference to a profile by assigning an URI to reduce the load of bandwidth. The URI is the known as a profile repository. For most of the available mobile devices, the device profiles have been created by the manufacturers. Therefore, it is easy to store these profiles in a repository. As for devices that have no leagcy profile, the creation of a new one for each device is necessary. But this creation would lead to another issue in interoperability. It is inevitable for developers to deal with different versions of device profiles. Therefore, the mechanism for handling multiple profile vocabularies must be taken into consideration in DANs.

3. Location Service in DAN

Mobility is the defining feature of wireless devices. In the Internet, the Mobile IP protocol was designed to support a mobile host. This concept can be introduced as a location service in DANs that can make the management of end devices more efficient, especially in battlefield environments. It is not currently practical to equip each device with a Global Positioning System (GPS) due to the cost. Instead, we can make use of the available Internet protocol that support mobile communication to approximate the location of an end device. For example, the Session Initial Protocol (SIP) developed by Internet Engineering Task Force (IETF) can be implemented for location service. SIP is a signaling protocol for Internet conferencing, telephony, presence, events notification and instant messaging [25]. With such functionality, we can provide a framework which is capable of location acquisition. The system architecture is shown in Figure 23.
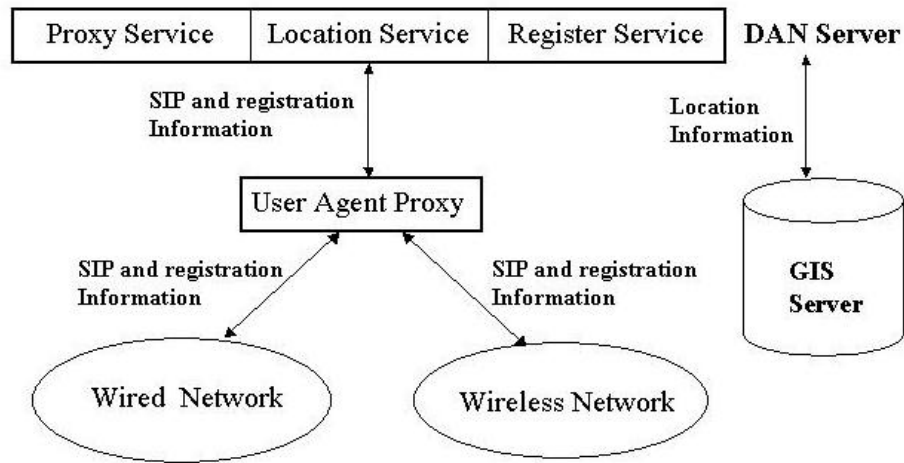
Figure 23.   SIP Framework

In the SIP protocol, the users logical location information can be transmitted during communication. Therefore, when a user enters a DAN network, the user location information can be mapped to a Geographic Information Server (GIS) to approximate the physical location.

4.   Performance Evaluation in DAN

Efficiency is the primary concern when delivering device profiles and adapting content in DANs. The purpose of DAN is to create an environment that can utilize the limited network bandwidth, especially in wireless networks. In fact, there are many factors that may influence the performance of a network. From the DAN perspective, the major factors that may influence the efficiency include transport protocol and intermediate user agent proxies. When a client sends the device profile to a server, the information should be encapsulated in a protocol header. The overhead of a packet that includes such a header must be taken into account when measuring the performance of a network. After the packet is delivered, it is the user agent proxy's responsibility to process and resolve the packet efficiently. The performance can be evaluated based upon the two testing points. But sometimes the performance is unpredictable when we use an end device to do the proxy function in an Ad-Hoc network. Therefore, another evaluation mechanism must be provided when developing a DAN network.

APPENDIX A

Component: HardwarePlatform

Attribute	Description	Type	Static/ Dynamic	Sample Value
ExternalPower	Indicates whether the device is currently connected to AC Power or any other power source, such as a cigarette lighter in a car. "Yes" means External Power is ON. "No" means External Power is OFF.	Boolean	Dynamic	"Yes," "No"
BatteryChargeStatus	Gives the current status of the battery as the percentage of battery charge remaining.	Number	Dynamic	"10," 55," "80"
BatteryLifetime	Full lifetime of fully charged battery (in seconds).	Number	Static	28800
BatteryLifetime Remaining	Remaining lifetime of battery, in seconds	Number	Dynamic	"1200"
BackupBattery ChargeStatus	Gives the current status of the backup battery as the percentage of battery charge remaining.	Number	Dynamic	"10," 55," "80"
BackupBattery Lifetime	Full lifetime of fully charged backup battery (in seconds).	Number	Static	28800
BackupBattery LifetimeRemaining	Remaining lifetime of backup battery, in seconds.	Number	Dynamic	"1600"
NumberOfProcessors	Total number of applications and communications processors in the device.	Number	Static	"1," "2 "
CPURevision Applications	Stepping of the Application Processor. The UAProf "CPU" attribute is to be used to	Literal	Static	"A0," "A1," "B1"

	specify the name and model number of the processor, such as "PXA 250" or "SA 1110."			
CPUFrequency	Current core clock frequency of the Applications Processor, in MHz.	Literal	Dynamic	"353.95," "200," "100"
CPUFrequency Maximum	Maximum core clock frequency of the Applications Processor, in MHz.	Number	Static	"200," "400"
CPUVoltage	Current voltage of the Applications Processor (in Volts)	Literal	Dynamic	"1.5," "1.0"
CommProcessor	Name and model number of the communication processors.	Literal	Static	"CXY123"
CommProcessor Revision	Stepping of the Communications Processor.	Literal	Static	"A0," "A1," "B1," "C0"
DynamicFrequency ChangeCapable	Indicates if the platform has Dynamic Frequency Management capability or not.	Boolean	Static	"Yes," "No"
HighConstrast DisplayMode	Indicates if high contrast display feature is available and on.	Boolean	Dynamic	"Yes," "No"
BacklightOn	Indicates if the display backlight is ON or OFF.	Boolean	Dynamic	"Yes," "No"
SIMType	Type of the Subscriber Identity Module in the device.	Literal	Static	"SIM," "USIM"
SIMToolkitVersion	Version number of the SIM Toolkit installed, if any. A version number of 0 indicates that SIM Toolkit is not installed. SIM Toolkit Vendor name can be included, if needed.	Literal	Static	"0," "2.1"
AvailableExpansion Slots	Lists the types of expansion slots available	Literal (Bag)	Static	"PCMCIA," "Compact

				Flash," "MMC"
ExpansionCards Inserted	Identifies the cards currently inserted, such as PCMCIA 802.11, CompactFlash 802.11, PCMCIA GPRS, etc.	Literal (Bag)	Dynamic	"PCMCIA 802.11," "CF 802.11," "CF Memory," "PCMCIA GPRS," "MMC," "CDPD"
CommunicationPorts	Lists all the available means for communication with a host computer, such as Serial Communications port, USB and IrDA.	Literal (Bag)	Static	"Serial," "USB Host," "USB Client," "IrDA," "Ethernet"
DeviceTemperature	Indicates the temperature of CPU.	Number	Dynamic	"50,""80"

Component: SoftwarePlatform

Attribute	Description	Type	Static/ Dynamic	Sample Value
CommProcessorOS Name	Name of the communications processor's operating system.	Literal	Static	"Smartphone 2002," "Nucleus"
CommProcessorOS Vendor	Vendor of the communications processor's operating system.	Literal	Static	"Microsoft," "Symbian"
CommProcessorOS Version	Version of the communications processor's operating system.	Literal	Static	"1.0," "2.5"
TotalProgram Memory	Total memory in MB that can be utilized by runtime programs.	Number	Dynamic	"32," "64," "128"
AvailableProgram Memory	Free program memory in MB that is currently	Literal	Dynamic	"12.45," "64," "128"

	available to runtime programs, not including the video frame buffer if present.			
FrameBufferSize	Size of the video frame buffer in KB.	Number	Dynamic	"512," "256"
TotalStorageMemory	Size of the total non-persistent file storage memory space on the device, in MB.	Number	Dynamic	"64," "256"
AvailableStorage Memory	Size of the available non-persistent file storage memory space on the device, in MB.	Literal	Dynamic	"63.24," "30.16"
TotalRemovable StorageMemory	Total removable storage card memory in MB.	Number	Dynamic	"16," "32"
AvailableRemovable StorageMemory	Available removable storage card memory in MB.	Literal	Dynamic	"15.6," "32"
TotalPersistent Memory	Total flash or other form of persistent file storage memory on the device, in MB. This memory persists across a total power loss, such as a dead battery or hard reset.	Number	Dynamic	"32," "64," "128"
AvailablePersistent Memory	Available flash or other form of persistent file storage memory on the device, in MB. This memory persists across a total power loss, such as a dead battery or hard reset.	Literal	Dynamic	"12.45," "64," "128"
PersistentMemory Manager	Type of persistent memory manager software.	Literal	Dynamic	"PSM," "FDI," "VFM"
PersistentMemory ManagerVersion	Version of persistent memory manager software.	Literal	Dynamic	"1.0," "2.0"
PersistentMemory XIP	Specifies whether the software platform supports Execute-In-Place or not.	Boolean	Dynamic	"Yes," "No"

MessagingServices	List of messaging capabilities (i.e., SMS, ESMS, MMS).	Literal (Bag)	Dynamic	"SMS," "MMS"

Component: NetworkCharacteristics

Attribute	Description	Type	Static/ Dynamic	Sample Value
CurrentBearerSignal Strength	The signal strength as a level between 0-100.	Literal	Dynamic	"0," "60," "100"
CurrentBearer MaximumBitRate	The maximum bit rate in Kbps for the current bearer service.	Literal	Dynamic	"56.6," "10000"
CurrentBearerActual BitRate	The current actual bit rate in Kbps for the current bearer service.	Literal	Dynamic	"26.4," "100"
SupportedBearer MaximumBitRates	The maximum reported bit rate in Kbps for each supported bearer service. The bearers in this list may or may not have an active connection.	Literal (Bag)	Static	"56.6," "1600," "28.8"
ActiveBearers	A list of bearers from "SupportedBearers" that currently have an active connection to a router or gateway device.	Literal (Bag)	Dynamic	"GPRS," "SMS," "802.11"
ActiveBearer Addresses	The address, for each of the bearers listed in "ActiveBearers" in the appropriate format, IP or UMTS.	Literal (Bag)	Dynamic	"145.19.22.14," "555-555-6262," "192.168.12.14"
ActiveBearerActual BitRates	The list of bit rates supported by the bearers listed in "SupportedBearers," the items should match one for one with the list given in "ActiveBearers"	Literal (Bag)	Dynamic	"56.6," "1600," "28.8"
ConnectedToHost	Indicates if the device is currently connected to a host computer through USB, Bluetooth, or by any other means.	Boolean	Dynamic	"Yes," "No"

CellID	Identifies the service bearer cell that the device is in at the current time.	Literal	Dynamic	"2001," "100"

Component: BrowserUA

Attribute	Description	Type	Static/ Dynamic	Sample Value
VoiceXMLCapable	Indicates whether the browser has Voice XML capability.	Boolean	Static	"Yes," "No"
TextToSpeech Capable	Indicates whether the browser has Text To Speech (TTS) capability.	Boolean	Static	"Yes," "No"
SpeechRecognition Capable	Indicates whether the browser has Speech Recognition capability.	Boolean	Static	"Yes," "No"

APPENDIX B

INTEL® PCA profile example in RDF:

```xml
<?xml version="1.0" ?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330#"
xmlns:pca="http://developer.intel.com/pca/developernetwork/devsupport/pca_schema/2002_01#">
<rdf:Description rdf:ID="MyPCADeviceProfile">
<prf:component>
<rdf:Description rdf:ID="HardwarePlatform">
<rdf:type rdf:resource=
"http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010330#HardwarePlatform"/>
<prf:ScreenSize>80x100</prf:ScreenSize>
<prf:Model>S100</prf:Model>
<prf:Vendor>Intel</prf:Vendor>
<prf:CPU>PXA250</prf:CPU>
<prf:InputCharSet>
<rdf:Bag>
<rdf:li>ISO-8859-1</rdf:li>
<rdf:li>US-ASCII</rdf:li>
<rdf:li>UTF-8</rdf:li>
<rdf:li>ISO-10646-UCS-2</rdf:li>
</rdf:Bag>
</prf:InputCharSet>
<prf:ScreenSizeChar>15x20</prf:ScreenSizeChar>
<prf:BitsPerPixel>8</prf:BitsPerPixel>
<prf:ColorCapable>Yes</prf:ColorCapable>
<prf:TextInputCapable>Yes</prf:TextInputCapable>
<prf:ImageCapable>Yes</prf:ImageCapable>
<prf:Keyboard>PhoneKeypad</prf:Keyboard>
<prf:NumberOfSoftKeys>0</prf:NumberOfSoftKeys>
<prf:OutputCharSet>
<rdf:Bag>
<rdf:li>ISO-8859-1</rdf:li>
<rdf:li>US-ASCII</rdf:li>
<rdf:li>UTF-8</rdf:li>
<rdf:li>ISO-10646-UCS-2</rdf:li>
</rdf:Bag>
</prf:OutputCharSet>
<prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
<prf:StandardFontProportional>Yes</prf:StandardFontProportional>
<prf:PixelsAspectRatio>1x1</prf:PixelsAspectRatio>
<pca:BatteryLifetime>28800</pca:BatteryLifetime>
<pca:NumberOfProcessors>2</pca:NumberOfProcessors>
<pca:CPURevision>B0</pca:CPURevision>
<pca:CPUFrequency>200</pca:CPUFrequency>
<pca:CPUFrequencyMaximum>400</pca:CPUFrequencyMaximum>
<pca:CommProcessor>CXY123</pca:CommProcessor>
<pca:CommProcessorRevision>B1</pca:CommProcessorRevision>
<pca:HighContrastDisplayMode>Yes</pca:HighContrastDisplayMode>
```

```xml
<pca:AvailableExpansionSlots>
<rdf:Bag>
<rdf:li>PCMCIA</rdf:li>
<rdf:li>Compact Flash</rdf:li>
<rdf:li>MMC</rdf:li>
</rdf:Bag>
</pca:AvailableExpansionSlots>
<pca:ExpansionCardsInserted>
<rdf:Bag>
<rdf:li>PCMCIA GPRS</rdf:li>
<rdf:li>CF 802.11</rdf:li>
</rdf:Bag>
</pca:ExpansionCardsInserted>
</rdf:Description>
</prf:component>
<prf:component>
<rdf:Description
rdf:ID="SoftwarePlatform">
<rdf:type rdf:resource=
"http://www.wapforum.org/profiles/UAPROF/ccppsch
ema-20010330#SoftwarePlatform"/>
<prf:OSName>PocketPC 2002</prf:OSName>
<prf:OSVendor>Microsoft</prf:OSVendor>
<pca:CommProcessorOSName>Nucleus</pca:CommProcessorOSName>
<pca:AvailableProgramMemory>128</pca:AvailableProgramMemory>
<prf:JVMVersion>Geode/1.0</prf:JVMVersion>
</rdf:Description>
</prf:component>
<prf:component>
<rdf:Description
  rdf:ID="NetworkCharacteristics">
<rdf:type rdf:resource=
"http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010330#NetworkCharacteristics"/>
<prf:SupportedBearers>
<rdf:Bag><rdf:li>GSM</rdf:li>
        <rdf:li>GPRS</rdf:li>
</rdf:Bag>
</prf:SupportedBearers>
<pca:CurrentBearerMaximumBitRate>56.6</pca:CurrentBearerMaximumBitRat
e>
</rdf:Description>
</prf:component>
<prf:component>
<rdf:Description rdf:ID="BrowserUA">
<rdf:type rdf:resource=
"http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330#BrowserUA"/>
<prf:BrowserName>Pocket IE</prf:BrowserName>
<prf:HtmlVersion>3.2</prf:HtmlVersion>
<prf:FramesCapable>Yes</prf:FramesCapable>
<prf:TablesCapable>Yes</prf:TablesCapable>
<prf:JavaAppletEnabled>No</prf:JavaAppletEnabled>
<prf:JavaScriptEnabled>Yes</prf:JavaScriptEnabled>
<prf:JavaScriptVersion>1.1</prf:JavaScriptVersion>
<pca:TextToSpeechCapable>No</pca:TextToSpeechCapable>
```

```
<pca:VoiceXMLCapable>Yes</pca:VoiceXMLCapable>
</rdf:Description>
</prf:component>
</rdf:Description>
</RDF>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1.   Stephane Boyera and Rhys Lewis, "An Introduction to Device Independence" [http://www.w3.org/2001/di/IntroToDI.html], October 2004

2.   Mark H. Butler, "Current Technologies For Device Independence, HPL-2001-83" [http://www.hpl.hp.com/personal/marbut/currTechDevInd.htm - _Toc510867875], October 2004

3.   Su Wen, and others, "Towards Device-Aware Networks," paper presented at the conference at Naval Postgraduate School, July 2004

4.   Graham Klyne, and others, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0" [http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/], October 2004

5.   Hidetaka Ohto and Johan Hjelm, "CC/PP exchange protocol based on HTTP Extension Framework" [http://www.w3.org/1999/06/NOTE-CCPPexchange-19990624], October 2004

6.   Wireless Application Group (WAG) SPEC-UAProf-19991110, *User Agent Profile Specification*, 10 November 1999

7.   Microsoft Corporation White Paper, *Understanding Universal Plug and Play*, June 2000

8.   Ericsson, and others, *SyncML Representation Protocol, version 1.0.1*, 15 June 2001

9.   Bruce Martin and Bashar Jano, "WAP Binary XML Content Format " [http://www.w3.org/TR/wbxml/], October 2004

10.   Wireless Application Group (WAG) WAP-248-UAProf-20011020-a, *User Agent Profile Specification*, 20 October 2001

11.   Mark H. Butler, "CC/PP and UAProf: Issues, Improvements and Future Directions, HPL-2002-35" [http://www.hpl.hp.com/techreports/2002/HPL-2002-35.pdf], October 2004

12.   Intel PCA Developer Network – Overview, [http://www.intel.com/pca/developernetwork/overview/index.htm],October 2004

13.   Intel 251604-001, *Intel PCA Device Profile Design Guide, Revision 1.0* , 8 August 2002

14. Mark H. Butler, "Implementing Content Negotiation Using CC/PP and WAP UAProf, HPL-2002-190" [http://www.hpl.hp.com/techreports/2001/HPL-2001-190.pdf], October 2004

15. Frank Manola and Eric Miller, "RDF Primer, W3C Recommendation 10 February 2004 " [http://www.w3.org/TR/rdf-primer/], October 2004

16. Charles Smith and Mark H. Butler, "Validating CC/PP UAProf Profiles, HPL-2002-268" [http://www.hpl.hp.com/techreports/2002/HPL-2002-268.pdf],October 2004

17. James Clark, "XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999" [http://www.w3.org/TR/xslt], October 2004

18. Mark H. Butler, "Using Capability Classes to Classify and Match CC/PP and UAProf Profiles" [http://www.hpl.hp.com/personal/marbut/capClass.htm], October 2004

19. James Clark and Steve DeRose, "XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999" [http://www.w3.org/TR/xpath], October 2004

20. W3C, "CC/PP Information Page" [http://www.w3.org/Mobile/CCPP/], October 2004

21. Mark H. Butler, "Device Independence and The Web, HPL-2002-249" [http://www.hpl.hp.com/research/papers/2003/device_independence.pdf], October 2004

22. Mark H. Butler, "Deli: A Delivery context Library for CC/PP and UAProf, HPL-2001-260" [http://www.hpl.hp.com/personal/marbut/DeliUserGuideWEB.htm], October 2004

23. Wireless Application Group (WAG) WAP-229-HTTP-20010329-a, *Wireless Profiled HTTP*, 29 March 2001

24. The Apache Cocoon Project [http://cocoon.apache.org/], October 2004

25. Session Initial Protocol (SIP) [http://www.cs.columbia.edu/sip/], October 2004

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman
   Information Sciences Department
   Naval Postgraduate School
   Monterey, California

4. Professor Singh Gurminder
   Naval Postgraduate School
   Monterey, California

5. Jonn Gibson
   Naval Postgraduate School
   Monterey, California

6. Shang-Yuan Tsai
   Chung-Shan Institute of Science and Technology
   Taoyuan, Taiwan